Submask Enumeration

Enumerating all submasks of a given mask

Given a bitmask m, you want to efficiently iterate through all of its submasks, that is, masks s in which only bits that were included in mask m are set.

Consider the implementation of this algorithm, based on tricks with bit operations:

```
int s = m;
while (s > 0) {
 ... you can use s ...
 s = (s-1) & m;
}
```

or, using a more compact `for` statement:

```
for (int s=m; s; s=(s-1)&m)
 ... you can use s ...
```

In both variants of the code, the submask equal to zero will not be processed. We can either process it outside the loop, or use a less elegant design, for example:

```
for (int s=m; ; s=(s-1)&m) {
 ... you can use s ...
 if (s==0)  break;
}
```

Let us examine why the above code visits all submasks of m, without repetition, and in descending order.

Suppose we have a current bitmask s, and we want to move on to the next bitmask. By subtracting from the mask s one unit, we will remove the rightmost set bit and all bits to the right of it will become 1. Then we remove all the "extra" one bits that are not included in the mask m and therefore can't be a part of a submask. We do this removal by using the bitwise operation `(s-1) & m`. As a result, we "cut" mask $s-1$ to determine the highest value that it can take, that is, the next submask after s in descending order.

Thus, this algorithm generates all submasks of this mask in descending order, performing only two operations per iteration.

A special case is when $s=0$. After executing $s-1$ we get a mask where all bits are set (bit representation of -1), and after `(s-1) & m` we will have that s will be equal to m. Therefore, with the mask $s=0$ be careful — if the loop does not end at zero, the algorithm may enter an infinite loop.

Iterating through all masks with their submasks. Complexity $O(3^n)$

In many problems, especially those that use bitmask dynamic programming, you want to iterate through all bitmasks and for each mask, iterate through all of its submasks:

```
for (int m=0; m<(1<<n); ++m)
    for (int s=m; s; s=(s-1)&m)
 ... s and m ...
```

Let's prove that the inner loop will execute a total of $O(3^n)$ iterations.

First proof: Consider the i-th bit. There are exactly three options for it:

1. it is not included in the mask m (and therefore not included in submask s),
2. it is included in m, but not included in s, or
3. it is included in both m and s.

As there are a total of n bits, there will be 3^n different combinations.

Second proof: Note that if mask m has k enabled bits, then it will have 2^k submasks. As we have a total of $\binom{n}{k}$ masks with k enabled bits (see binomial coefficients), then the total number of combinations for all masks will be:

$$\sum_{k=0}^{n} \binom{n}{k} \cdot 2^k$$

To calculate this number, note that the sum above is equal to the expansion of $(1+2)^n$ using the binomial theorem. Therefore, we have 3^n combinations, as we wanted to prove.

Practice Problems

- Atcoder - Close Group
- Codeforces - Nuclear Fusion
- Codeforces - Sandy and Nuts
- Uva 1439 - Exclusive Access 2
- UVa 11825 - Hackers' Crackdown